Environments

Introduction

In the Play+ ecosystem, applications must seamlessly adapt to varying runtime contexts—from local development to staging and production. playenv supports this through environment-aware configuration , delivering a secure, consistent, and reliable way to manage environment-specific variables.

A robust environment management system enhances both security and maintainability . playenv ensures that API keys, feature flags, and contextual settings are loaded correctly without being hardcoded.

Package Info

Distribution Path	Description
Golden Path	Pre-installed at /system/play.env.ts
Uplift Path	npm install @playplus/env

Folder Reference

Environment variables are managed using .env files at the project root:

File / Directory	Purpose & Guidelines
.env	Default values for all environments. Committed to version control.
.env.local	Local overrides for development. Must not be committed.
.env.development	Optional environment-specific file.
.env.production	Used in production deployment.
/system/play.env.ts	Core helper exposing type-safe access to environment variables.

File / Directory	Purpose & Guidelines
/system/play.env.service.ts	Angular service wrapper for environment management.

Pillar Alignment

Pillar	Alignment Description
Adaptive	Enables dynamic behavior across environments (e.g., API endpoints, logging levels).
Intuitive	Offers a simple, type-safe get() interface with built-in coercion.
Distinct	Promotes consistent, repeatable config practices across Play+ projects.
Inclusive	Reduces cognitive load for new or experienced developers.
Engaging	Encourages best practices like secret segregation and fallback defaults.

Helper Overview

playenv is a secure, type-safe abstraction over process.env . It simplifies and automates:

- Variable Loading : Automatically reads from .env , .env.[env] , and .env.local in priority order.
- Override Strategy : .env.local > .env.[env] > .env
- Type Parsing : Coerces values to string , number , or boolean .
- Validation : Optionally checks required variables at startup.
- Security : Separates secrets into .env.local , helping avoid accidental commits.
- Cross-Platform : Works in both browser and Node.js environments (Angular/SSR).

Configuration Options

All configuration is done via .env files. No separate JSON config is needed.

Variable Name	Example Value	Description
NODE_ENV	"development"	Current runtime environment.
API_URL	"https://api.playplus.io"	Backend API base URL.
FEATURE_FLAG_CLIENT_KEY	"sdk-123-abc"	Client-side key for feature flag service.
ENABLE_ANALYTICS	true	Boolean toggle for analytics scripts.
LOG_LEVEL	"debug"	Logging level for the application.
LOG_TRANSPORT	"console"	Log transport method.
LOG_API_URL	"https://logs.api.com"	Remote logging API URL.
LOG_API_KEY	"log-key-123"	Remote logging API key.
APP_NAME	"play-angular-seed"	Application name.
HOSTNAME	"localhost"	Application hostname.
AUTH_STORAGE	"localStorage"	Authentication storage method.
AUTH_TOKEN_NAME	"auth_token"	Authentication token name.
AUTH_SCHEME	"Bearer"	Authentication scheme.
CSRF_COOKIE_NAME	"csrf_token"	CSRF cookie name.
CSRF_HEADER_NAME	"X-CSRF-Token"	CSRF header name.

Helper Methods

Core Methods

Method Name	Description	Signature
get	Get an environment variable with type coercion and fallback.	get<T = string>(key: string, defaultValue?: T): T

Method Name	Description	Signature
validate	Validate that required environment variables are set.	validate(requiredKeys: string[]): void
all	Expose all loaded environment variables (for debugging only).	all(): Record<string, string | undefined>

Type Coercion Logic:

- If T is boolean , parses 'true' / 'false' and '1' / '0'
- If T is number , attempts parseFloat
- Returns defaultValue if key is undefined
- Throws error if required key is missing and no default provided

Angular Service Methods

The PlayEnvService provides additional Angular-specific functionality:

Method Name	Description	Return Type
setConfig	Set environment configuration.	void
get	Get configuration value by key.	any
isProduction	Check if running in production.	boolean
isDevelopment	Check if running in development.	boolean
isDebug	Check if debug mode is enabled.	boolean
getApiUrl	Get the API URL.	string
getAppName	Get the application name.	string
getVersion	Get the application version.	string

Usage Examples

Angular: Config Service (Implemented)

Angular: PlayEnvService Usage

Debugging Example

Git Configuration

The project includes proper .gitignore configuration for environment files:

Note : The environment files are commented out in the current .gitignore to allow for development setup. In production projects, these should be uncommented.

Developer Checklist

Before You Commit:

- Added all non-secret keys to .env with sensible defaults?
- Stored secrets in .env.local (not .env)?
- Confirmed .env.local is in .gitignore ?
- Using playenv.get() instead of process.env ?
- Provided fallback defaultValue in get() calls?
- (Optional) Configured required key validation?
- Used type coercion for boolean and number values?
- Validated environment variables at app startup?

Why We Created This Helper

Accessing configuration via process.env is common but problematic:

- Not type-safe — values are strings by default
- Scattered — fallback logic lives across many files
- Insecure — secrets can be committed by mistake
- Opaque — missing keys cause silent failures
- Platform-specific — different behavior in browser vs Node.js

playenv solves all of these:

- Centralized, predictable, validated config access
- Built-in type safety with coercion
- Strong separation of code and secrets
- Cross-platform compatibility (browser + Node.js)
- Clear error messages for missing required variables

Integration with Other Play+ Systems

Logging Integration

Security Integration

Server Configuration

Closing Note

Environment awareness is not just a backend concern — it's a foundation for resilient frontends too. playenv empowers developers to ship confidently across contexts, knowing that their applications are secure, reliable, and configuration-smart.

It's another way Play+ helps you focus on what matters — the experience.